MCU 4200 Series Remote Management API

This specification may be subject to change without notice

Codian Confidential

Version 2.0.1

April 2006

Contents

Introduction 3
1.1 HTTP 3
1.2 XML-RPC 3
1.3 Authentication 3
1.4 Message Flow 3
2 Messages 6
Authentication 6
2.1 conference.create 6
2.2 conference.modify 8
2.3 conference.destroy 9
2.4 conference.end 10
2.5 conference.participant.add 11
2.6 conference.participant.remove 12
2.7 conference.participant.modify 13
2.8 participant.connect 13
2.9 participant.disconnect 14
2.10 device.query 14
2.11 participant.enumerate 14
2.12 conference.enumerate 16
2.13 autoAttendant.enumerate 18
2.14 autoAttendant.destroy 18
2.15 participant.fecc 19
2.16 participant.message 19
2.17 participant.diagnostics 19
3 Deprecated messages 21
3.1 system.query 21
3.2 conference.query 21
4 Required user privileges 23
5 Fault Codes 24
6 Participant disconnect reasons 25
7 References 26
Appendix A - Conference Layouts 27
Appendix B - Linking conferences across MCUs 30
B.1 Example message 1 - creating conference "linked1" on "mcu1" 30
B.2 Example message 2 - creating conference "linked2" on "mcu2" 31
B.3 Example message 3 - calling into "linked2" from "linked1" 31
B.4 Example message 4 - setting the new "linked2" participant to use a full screen view layout 32
B.5 Message responses 33

Revision History

Version	Date	Author	Comments
1.0	20 Apr 2005	MKL, AP	Initial release detailing the API support in version 1.2(1) of the MCU 4200 series software.
2.0	6 Feb 2006	MKL, AP, AB	Draft including the API enhancements to be released in version 1.4(1) of the MCU 4200 series software.
2.0.1	4 Apr 2006	AP, MKL	Corrections and additional information. Updated Appendix A, removed outdated Appendix B

Copyright © Codian Ltd. 2004 - 2006

Introduction

The Codian MCU 4200 series products are controllable via messages sent using the XML-RPC protocol.

Please note that the Management API option key must be loaded before an MCU 4200 series device will respond to API commands.

XML-RPC is a simple protocol for remote procedure calling using HTTP as the transport and XML as the encoding. It is designed to be as simple as possible, whilst allowing complex data structures to be transmitted, processed and returned. XML-RPC has no platform or software dependence. XML-RPC was chosen over SOAP due to its simplicity.

The interface is stateless. Currently there is no mechanism for the Codian MCU to call back the controlling application and therefore the controlling application must poll the MCU for status as required. A future enhancement *may* provide a mechanism for signaling MCU status changes to the controlling application.

1.1 HTTP

The Codian MCU expects to receive HTTP communication over TCP/IP connections to port 80. The HTTP messages should be “POST”s to the URL “/RPC2”.

The MCU implements HTTP/1.1 as defined by RFC 2616 [2].

1.2 XML-RPC

For the background and details of XML-RPC, please refer to the specification [1].

In this implementation, all parameters and return values are part of a <struct> and are all explicitly named. For example, the “device.query” method returns the current time value as a structure member named ‘currentTime’ rather than as a single value of type <dateTime.iso8601>.

1.3 Authentication

In order to manage the MCU, the controlling application must authenticate itself as a user with administrator privileges. Accordingly, each message contains a user name and password.

1.4 Message Flow

An application can create and manage conferences by sending command messages to the MCU. For each command sent, the MCU will respond with a message indicating success or failure. The response message may also contain data requested.

Command messages are sent in XML format. For example, the following message will schedule a conference to begin at 10:45 on 18 February 2005 and last for one hour:

```
POST /RPC2 HTTP/1.1
User-Agent: Frontier/5.1.2 (WinNT)
Host: 10.2.1.100
Content-Type: text/xml
Content-length: 713

<?xml version="1.0"?>
<methodCall>
<methodName>conference.create</methodName>
<params>
<param>
<value>
<struct>
```

Copyright © Codian Ltd. 2004 - 2006

```
<member>
<name>authenticationUser</name>
<value>
<string>api_test</string>
</value>
</member>
<member>
<name>authenticationPassword</name>
<value>
<string>123456</string>
</value>
</member>
<member>
<name>conferenceName</name>
<value>
<string>Meeting 1</string>
</value>
</member>
<member>
<name>startTime</name>
<value>
<dateTime.iso8601>20050218T10:45:00</dateTime.iso8601>
</value>
</member>
<member>
<name>durationSeconds</name>
<value>
<int>3600</int>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodCall>
```

If the command was successful, the MCU will send a success response. For example, in response to a successful conference.create message, the MCU will return:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 240

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value>
<struct>
<member>
<name>status</name>
<value>
<string>operation successful</string>
</value>
</member>
</struct>
</value>
```

Copyright © Codian Ltd. 2004 - 2006

```
</param>
</params>
</methodResponse>
```

If the command fails, the MCU will send a fault response. For example, in response to a “conference.create” message where the conference name is not unique, the MCU will return:

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Content-Length: 411

<?xml version="1.0"?>
<methodResponse>
<fault>
  <value>
    <struct>
      <member>
        <name>faultCode</name>
        <value>
          <int>2</int>
        </value>
      </member>
      <member>
        <name>faultString</name>
        <value>
          <string>duplicate conference name</string>
        </value>
      </member>
    </struct>
  </value>
</fault>
</methodResponse>
```

The complete list of command messages, their required and optional parameters, and the expected responses are detailed in section 2. The possible fault codes are listed in section 5. Appendix B contains examples of some messages and their corresponding responses.

Copyright © Codian Ltd. 2004 - 2006

2 Messages

Authentication

All messages must contain a user name and password as follows:

Parameter	Type	Comments
authenticationUser	string	Name of a user with sufficient privilege for the operation being performed. The name is case sensitive.
authenticationPassword	string	The corresponding user's password.

Participant identification parameters

The following parameters appear in the majority of messages, and are used to identify a specific participant on which operations are to be performed.

Parameter	Type	Description
participantName	string	This is an "internal" name, and hence not necessarily related to any name configured on an endpoint itself. Within the scope of a particular conference or auto attendant, the combination of "participantType", "participantProtocol" and "participantName" is always unique.
participantProtocol	string	Used in conjunction with "participantName" to uniquely identify a participant within a connection. Typically these should be treated as opaque values, but the current possibilities are: for "participantProtocol": h323 – the H.323 protocol vnc – a VNC connection (e.g. remote desktop) for "participantType": ad_hoc – this participant called into the MCU by_address – fully-specified participant by_name – MCU-configured endpoint API-created participants (i.e. those originated by "conference.participant.add") will be of type "by_address".
participantType	string	
conferenceName	string	Unique conference name – the conference name space is shared between API-generated conferences and all other ad hoc and scheduled MCU conferences.
autoAttendantUniqueId	string	If the participant in question is connected to an auto attendant rather than a conference, this field contains a unique identifier for that auto attendant.

When modifying or querying parameters for a specific endpoint, "participantName", "participantProtocol" and "participantType" parameters are supplied, along with either a "conferenceName" or an "autoAttendantUniqueId". However, as the API can only create H.323 participants of type "by_address", if the protocol and type fields are absent then they are assumed to be "h323" and "by_address".

2.1 conference.create

Copyright © Codian Ltd. 2004 - 2006

Parameter	Type	Comments
conferenceName	string (<32 chars)	Name of the conference to be created. The conference creation will fail if the name is not unique.
numericId (optional)	string (<32 chars)	Numeric identifier of the conference
conferenceId	string (< 32 chars)	Deprecated alternative for “numericId”
registerWithGatekeeper (optional)	boolean	Register the conference’s “numericId” with the gatekeeper
startTime (optional)	dateTime.iso8601	If you do not specify a time the conference will start immediately
durationSeconds (optional)	int	The length of each repeating conference instance, in seconds. If this parameter is absent, or set to “0”, the conference is permanent.
endTime (optional)	dateTime.iso8601	If you do not specify an end time then the conference will be permanent (except if it is explicitly deleted). **This parameter is deprecated and present for backward compatibility reasons only. Application code should use “durationSeconds” in preference.**
pin (optional)	string (< 32 chars)	If present, this is the string of numeric digits that people need to enter to join the conference.
description(optional)	string (< 32 chars)	
multicastStreamingEnabled(optional)	boolean	
unicastStreamingEnabled(optional)	boolean	
h239Enabled(optional)	boolean	
maximumAudioPorts	int	These fields set the limit on number of audio (audio only) and video (video + audio) ports for the conference. The “reserved” values are for port reservation mode, whereas the “maximum” figures apply to non-reserved mode (and will be absent if no limits have been configured).
maximumVideoPorts	int	
reservedAudioPorts	int	
reservedVideoPorts	int	
repetition(optional)	string	one of: none daily weekly everyTwoWeeks monthly
weekDay(optional)	string	Must be present if repetition is “monthly”; one of: monday tuesday wednesday thursday friday saturday sunday
whichWeek(optional)	string	Must be present if repetition is “monthly”; one of: first = first X of the month (where X is the day specified by weekday) second third

Copyright © Codian Ltd. 2004 - 2006

		fourth last = last X of the month
weekDays(optional)	string	Must be present if repetition is “weekly” or “everyTwoWeeks”; a comma separated string of weekdays from the following list: monday tuesday wednesday thursday friday saturday sunday eg. “monday,Wednesday,friday”
terminationType (optional)	string	one of: noTermination afterNRepeats endOnGivenDate
terminationDate (optional)	dateTime.iso8601	if terminationType is endOnGivenDate this is the day the conference repetition will end on
numberOfRepeats (optional)	int	if terminationType is afterNRepeats this is the number of repeats to end after

Conferences created through the management API will appear in the list of conferences accessible via the Web interface, and vice versa.

2.2 conference.modify

Parameter	Type	Comments
conferenceName	string (< 32 chars)	Name of the conference to modify
newConferenceName (optional)	string (< 32 chars)	If present, conference will be renamed to specified value
oldConferenceName	string (<32 chars)	**Deprecated conference renaming scheme – new code should use conferenceName and newConferenceName as above.**
conferenceName	string (<32 chars)	
numericId	string (< 32 chars)	Optional fields as per “conference.create” described above
conferenceId **(deprecated)**		
registerWithGatekeeper	boolean	
startTime	dateTime.iso8601	
durationSeconds	int	
endTime	dateTime.iso8601	
pin	string	
description	string	
multicastStreamingEnabled	boolean	
unicastStreamingEnabled	boolean	
h239Enabled	boolean	
reservedVideoPorts	int	
reservedAudioPorts	int	
maximumVideoPorts	int	
maximumAudioPorts	int	
repetition	string	
weekDay	string	
whichWeek	string	
weekDays	string	
terminationType	string	
terminationDate	dateTime.iso8601	
numberOfRepeats	int	

Conferences created through the management API will appear in the list of conferences accessible via the Web interface. The API can thus be used to modify conferences scheduled via the web interface, and vice versa.

2.3 conference.destroy

Parameter	Type	Comments
conferenceName	string	Name of the conference to be destroyed.

A conference can be destroyed at any time; that is, before the conference has begun, during the conference or after the conference has ended.

Copyright © Codian Ltd. 2004 - 2006

2.4 conference.end

Parameter	Type	Comments
conferenceName	string	Name of the conference to be ended.

A conference will remain in the list of conferences even after the conference has ended until conference.destroy is called.

The intent is that this message performs the same function as the "End conference" control on the MCU web interface. However, it is not currently implemented in any shipping version of Codian MCU 4200 series software.

Copyright © Codian Ltd. 2004 - 2006

2.5 conference.participant.add

Parameter	Type	Comments
conferenceName	string	The name of the conference to which to add the participant.
participantName	string (<32 chars)	The name of the participant to be added. This must be a unique value, i.e. not the same as any existing participant.
participantProtocol (optional)	string	If present, must be “h323” – this is the only protocol that the API can currently use.
participantType (optional)	string	If present, must be “by_address”.
address	string (< 32 chars)	The participant’s E.164 directory number, hostname or IP address.
gatewayAddress (optional)	string (< 32 chars)	IP address or hostname of an H.323 gateway
deferConnection (optional)	boolean	If true, means don’t call out to this participant immediately, but wait for a “[conference.]participant.connect” command.
All of the following parameters are optional, and control the conferencing behaviour of the MCU with respect to the endpoint in question, for example the maximum resolution of the video streams used, or whether the participant is able to control the conference view layout that they see.		
maxBitRateToMCU	int	The maximum bit rate to the MCU specified as kBit/s
maxBitRateFromMCU	int	The maximum bit rate from the MCU specified as kBit/s
displayNameOverrideStatus	Boolean	If true, means use the specified “displayNameOverrideValue” text as the participant’s display name during the conference.
displayNameOverrideValue	string (< 32 chars)	Value to use as the participant’s display name (if “displayNameOverrideStatus” set to true).
cpLayout	string	This sets the initial conference view layout for video sent to this participant. Refer to Appendix A for the full list of available layouts.
layoutControlEnabled	boolean	Controls whether this participant is able to change the conference view layout that they see; 1 (true) means that the participant can change the layout using FECC or DTMF, 0 (false) means that they cannot.
audioRxMuted	boolean	1 (true) means that audio from this participant will not be heard by other conference participants.
audioRxGainMode	string	one of: none – no extra gain applied automatic – automatic gain control applied fixed – fixed number of dBs of gain applied
audioRxGainMillidB	int	if audio gain mode “fixed”, this is the number of decibels of gain applied, multiplied by 1000, and can be a negative value
videoRxMuted	boolean	true means that video from this participant will not be seen by other conference participants
videoTxWidescreen	boolean	if true, means that video sent to this participant will be in a form suitable for a widescreen (16:9) display
videoTxMaxResolution	string	one of: cif 4cif max
videoRxMaxResolution	string	same as above

Copyright © Codian Ltd. 2004 - 2006

All participants in a conference must have a “participantName” that is unique to the conference but it need not be unique across all conferences.

Participants can be added before or during a conference.

If a “participantName” matches the name of an endpoint in the list of configured endpoints (via the Web interface, Home > Endpoint list) the two are treated as unrelated. This is because web interface named, configured, endpoints have the “participantType” value “by_name” whereas API participants are of type “by_address”.

2.6 conference.participant.remove

Parameter	Type	Comments
conferenceName	string	Participant identification as described above.
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

Copyright © Codian Ltd. 2004 - 2006

2.7 conference.participant.modify

Parameter	Type	Comments
conferenceName	string	Participant identification as described above.
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	
address	string	These parameters override the configured values, but have no effect on active participants.
gatewayAddress	string	
deferConnection	boolean	
maxBitRateToMCU	int	
maxBitRateFromMCU	int	
displayNameOverrideStatus	boolean	All of these parameters are optional, and override / change the values provided in the "conference.participant.add" call. As well as changing configured participant parameters, they also change active participant state, and so may have real-time effect.
displayNameOverrideValue	string	
cpLayout	string	
layoutControlEnabled	boolean	
audioRxMuted	boolean	
audioRxGainMode	string	
audioRxGainMillidB	int	
videoRxMuted	boolean	
videoTxWidescreen	boolean	
videoTxMaxResolution	string	
videoRxMaxResolution	string	
important (optional)	boolean	Specifies whether this participant is important. This setting only has an effect on active participants.

This method affects active conference participation as well as endpoint configuration. If the conference has not yet started, this operation effectively just changes the configured parameters as passed in "conference.participant.add". If the conference has started and the participant is connected, this method will change participants' layouts and audio mute status in real-time.

2.8 participant.connect

Parameter	Type	Comments
conferenceName	string	Participant identification as described above.
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

Used primarily for API-generated participants added with "deferConnection" set to true; this command causes the MCU to call out to such participants.

Copyright © Codian Ltd. 2004 - 2006 Codian Confidential

2.9 participant.disconnect

Parameter	Type	Comments
conferenceName	string	Participant identification as described above.
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

This call causes the MCU to tear down its connection to the specified participant, if such a connection exists. This is different from “participant.remove” above because in the case of configured participants it does not remove the configuration (thus allowing later re-connection with “participant.connect”) and in the case of ad hoc participants it does not remove the record of the previous connection.

2.10 device.query

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system’s current time.
restartTime	dateTime.iso8601	The date and time at which the system was last restarted.

2.11 participant.enumerate

Parameter	Type	Comments
enumerateID(optional)	string	The value returned by the last enumeration call, if omitted a new enumeration is started.

returns:

Response	Type	Comments
enumerateID(optional)	string	The value which should be used in the next call to get the next set of data

and an array called “participants” of structs which contain

Response	Type	Comments
participantName	string	Participant identification as described above.
participantProtocol	string	
participantType	string	
conferenceName	string	
autoAttendantUniqueId	string	
address	string	The address used to connect to the remote endpoint in question. Only returned when the address is known, or if the participant is configured via the API (which requires the address to be specified when added).
gatewayAddress	string	
deferConnection	boolean	
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only available after the participant has connected.
displayNameOverrideStatus	boolean	Indicates whether the displayName value is the result of being overridden
maxBitRateToMCU	int	As for “participant.add”; kbps values
maxBitRateFromMCU	int	
videoRxMaxResolution	string	As for “participant.add”; “cif”, “4cif” or “max”
videoTxMaxResolution	string	
callState	string	The value will be one of: - ‘dormant’ - ‘alerting’ - ‘connected’ - ‘disconnected’
connectTime	dateTime.iso8601	Only returned once the participant is connected. This value is always present if the call state is “connected”; it may or may not be defined for participants in the “disconnected” state, depending on whether they were ever connected.
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected
disconnectReason	string	Only returned after the participant has disconnected, one of the disconnect reason values given in table 5.
connectPending	boolean	true if sending a “participant.connect” command for this participant will cause either the initial connection to that endpoint (in the event that it was configured with “deferConnection” set) or a re-connection to that endpoint (in the event that it has disconnected).
audioRxCodec	string	
audioRxLost	int	
audioRxReceived	int	
audioTxCodec	string	
audioTxReportedLost	int	
audioTxSent	int	
audioRxMuted	boolean	
audioRxGainMode	string	
audioRxGainMillidB	int	
videoRxCodec	string	

Copyright © Codian Ltd. 2004 - 2006

videoRxLost	int	
videoRxReceived	int	
videoTxCodec	string	
videoTxReportedLost	int	
videoTxSent	int	
videoRxMuted	boolean	
videoTxWidescreen	boolean	
important	boolean	
activeSpeaker	boolean	true if this participant is currently the active speaker in the conference
layoutControlEnabled	boolean	
cpLayout	string	The configured layout behavior for this participant - see appendix A. This parameter will be present only for participants configured via the API.
currentLayout	int	Actual layout in use for the video stream being sent by the MCU to this participant. This parameter will not be present if the participant is in an auto attendant rather than a conference, nor if the MCU is not currentlytransmitting video to the participant in question.
callDirection	string	one of: incoming outgoing

Note: This participant information is returned for all participants added to the conference using the conference.participant.add method, even after they have disconnected. However, this information is only returned for other participants (i.e. those added via the Web interface or those who dialed into the conference) whilst they are connected but not after they have disconnected.

2.12 conference.enumerate

Parameter	Type	Comments
enumerateID (optional)	string	The value returned by the last enumeration call, if omitted a new enumeration is started.

returns:

Response	Type	Comments
enumerateID (optional)	string	The value which should be used in the next call to get the next set of data

and an array called “conferences” of structs which contain:

Copyright © Codian Ltd. 2004 - 2006

Response	Type	Comments
conferenceName	string	
conferenceType	string	one of: scheduled ad_hoc
conferenceActive	boolean	indicates whether conference is currently active
description	string	extra user-specified information about the conference
pin	string	the security PIN
numericId	string	
registerWithGatekeeper	boolean	
multicastStreamingEnabled	boolean	
unicastStreamingEnabled	boolean	
h239Enabled	boolean	
maximumAudioPorts	int	These fields set the limit on number of audio (audio only) and video (video + audio) ports for the conference. The “reserved” values are for port reservation mode, whereas the “maximum” figures apply to non-reserved mode (and will be absent if no limits have been configured).
maximumVideoPorts	int	
reservedAudioPorts	int	
reservedVideoPorts	int	
The following timing fields will be present for scheduled conferences only		
startTime	dateTime.iso8601	The time at which the conference started at or will start at.
durationSeconds	int	How long each repeating instance of the conference should last for – if absent, the conference is permanent.
repetition(optional)	string	one of: none daily weekly everyTwoWeeks monthly
weekDay(optional)	string	Must be present if repetition is monthly. One of: monday tuesday wednesday thursday friday saturday Sunday
whichWeek (optional)	string	Must be present if repetition is monthly one of: first = first X of the month (where X is the day specified by weekday) second third fourth last = last X of the month
weekDays (optional)	string	A comma separated string of weekdays from the

		following list: monday tuesday wednesday thursday friday saturday sunday eg. “monday,wednesday,friday” This field should be provided when repetition is weekly or everyTwoWeeks
terminationType(optional)	string	one of: noTermination afterNRepeats endOnGivenDate
terminationDate(optional)	dateTime.iso8601	if terminationType is endOnGivenDay this is the day the conference repetition will end on
The following timing values will be present for active conferences only		
activeStartTime	dateTime.iso8601	If the conference is currently active, these fields show the time span of the current activation. If the conference is permanent then “activeEndTime” will be absent.
activeEndTime	dateTime.iso8601	

2.13 autoAttendant.enumerate

Parameter	Type	Comments
enumerateID(optional)	string	The value returned by the last enumeration call, if omitted a new enumeration is started.

returns:

Response	Type	Comments
enumerateID(optional)	string	The value which should be used in the next call to get the next set of data

and an array called “autoAttendants” of structs which contain:

Response	Type	Comments
autoAttendantUniqueID	string	
startTime	dateTime.iso8601	The time at which the auto attendant was created.

2.14 autoAttendant.destroy

Parameter	Type	Comments
autoAttendantUniqueID	string	Identifier for the auto attendant to be destroyed.

Copyright © Codian Ltd. 2004 - 2006

2.15 participant.fecc

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	
direction	string	one of: up down left right

2.16 participant.message

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	
message	string	The string to send to the participant

2.17 participant.diagnostics

Parameter	Type	Comments
conferenceName	string	Participant identification as described above
autoAttendantUniqueId	string	
participantName	string	
participantProtocol	string	
participantType	string	

The method response returns the following:

Copyright © Codian Ltd. 2004 - 2006

Parameter	Type	Comments
videoTxFrameRate	int	
videoRxFrameRate	int	
videoRxFramesReceived	int	
videoTxChannelBitRate	int	
videoTxSelectedBitRate	int	
videoTxActualBitRate	int	
videoTxLimitReason	string	one of: notLimited, viewedSize, quality, aggregateBandwidth, flowControl, endpointLimitation,
videoRxChannelBitRate	int	
videoRxSelectedBitRate	int	
videoRxActualBitRate	int	
videoRxLimitReason	string	one of: notLimited, viewedSize, quality, aggregateBandwidth, flowControl, endpointLimitation,
videoTxWidth	int	
videoTxHeight	int	
videoTxInterlaced	boolean	
videoRxWidth	int	
videoRxHeight	Int	
videoRxInterlaced	boolean	

Copyright © Codian Ltd. 2004 - 2006

3 Deprecated messages

These messages are those that were supported in v1.0 of the MCU 4200 Series Management API, but have since been superseded in API v2.0.

3.1 system.query

This method is deprecated in favour of "device.query" and "conference.enumerate".

There are no parameters passed with this method call. The method response returns the following:

Parameter	Type	Comments
currentTime	dateTime.iso8601	The system's current time.
restartTime	dateTime.iso8601	The date and time at which the system was last restarted.

The method also returns a 'conferences' <array> of <struct> for each conference, where each <struct> contains the following parameters:

Parameter	Type	Comments
conferenceName	string	The name of the conference
numJoined	int	The number of participants that have ever joined the conference
numLeft	int	The number of participants that have ever left the conference. The difference between numJoined and numLeft gives the number of current participants.

Appendix **Error! Reference source not found.** contains an example of a response to this message showing several conferences.

Note that until conference.destroy is called for a particular conference, the conference will remain in the list of conferences even after the conference has ended.

3.2 conference.query

This method is deprecated in favour of "conference.enumerate" and "participant.enumerate".

Parameter	Type	Comments
conferenceName	string	The name of the conference of interest.

The method response returns the following:

Parameter	Type	Comments
startTime	dateTime.iso8601	The time at which the conference started at or will start at.
endTime	dateTime.iso8601	The time at which the conference will end; if the conference is permanent then this parameter is absent.
pin	string	The PIN

The method also returns a 'participants' <array> of <struct> for each conference, where each <struct> contains the following parameters:

Copyright © Codian Ltd. 2004 - 2006

Response	Type	Comments
participantName	string	The participant name supplied in the conference.participant.add message
displayName	string	The name used by the endpoint to identify itself. This may be different to the participantName. Only returned when the participant connected.
address	string	The address used to connect to the remote endpoint in question. Only returned when the address is known.
callState	string	The value will be one of: - 'dormant' - 'alerting' - 'connected' - 'disconnected'
connectTime	dateTime.iso8601	Only returned after the conference has begun
disconnectTime	dateTime.iso8601	Only returned after the participant has disconnected
disconnectReason	string	Only returned after the participant has disconnected

Note: This participant information is returned for all participants added to the conference using the conference.participant.add method, even after they have disconnected. However, this information is only returned for other participants (i.e. those added via the Web interface or those who dialed into the conference) whilst they are connected but not after they have disconnected.

Copyright © Codian Ltd. 2004 - 2006

4 Required user privileges

The following table summarises which users are permitted to perform which remote management operations.

Method	Valid users
device.query	Any user with administrator privileges
conference.enumerate	Any user with administrator privileges
participant.enumerate	Any user with administrator privileges
conference.create	Any user with conference creation abilities
conference.destroy	The owner of the conference, or a user with “conference creation and full control” or “administrator” privilege.
conference.modify	
conference.end	
conference.participant.add	Any user with “administrator” privilege, the owner of the conference, or a user with “conference creation and full control” rights.
autoAttendant.enumerate	Any user with administrator privileges
autoAttendant.destroy	
participant.remove	If the participant is connected to an auto attendant, administrator privileges are required. If the participant is in a conference, the user must be the owner of the conference or a user with “conference creation and full control” or “administrator” privilege.
participant.modify	
participant.diagnostics	
participant.fecc	
participant.message	
participant.connect	
participant.disconnect	
Deprecated methods	
system.query	Any user with administrator privileges.
conference.query	Any user with privilege “conference detail” or higher.
conference.participant.modify	As above for “participant.modify”
conference.participant.remove	As above for “participant.remove”

Copyright © Codian Ltd. 2004 - 2006

5 Fault Codes

Fault Code	Description
1	Method not supported
2	Duplicate conference name
3	Duplicate participant name
4	No such conference or auto attendant
5	No such participant
6	Too many conferences
7	Too many participants
8	No conference name or auto attendant id supplied
9	No participant name supplied
10	No participant address supplied
11	Invalid start time specified
12	Invalid end time specified
13	Invalid PIN specified
14	Unauthorized – invalid user id / password
15	Insufficient privileges – specified user id / password not valid for attempted operation
16	Invalid “enumerateID” value passed in “…enumerate” method invocation
17	Port reservation failure – “reservedAudioPorts” or “reservedVideoPorts” value too high
18	Duplicate numeric ID

Copyright © Codian Ltd. 2004 - 2006

6 Participant disconnect reasons

These are the possible values of the “disconnectReason” field in participant information responses:

Value	Description
unspecified	
unspecifiedError	
remoteTeardown	
localTeardown	
noAnswer	
moved	
rejected	
rejectedImmediately	
busy	
timeout	
gatekeeperError	
networkError	
protocolError	
dnsFailed	
destinationUnreachable	
gatekeeperEnded	
videoPortAllocationExceeded	
portAllocationExceeded	
disconnectAll	
incompatibleVncVersion	
failedToConnectToServer	
authenticationFailed	
serviceUnavailable	
capabilityNegotiationError	
messageQueueOverflow	
gatekeeperRequiredButAbsent	
noGatekeeperForDN	
localGatekeeperRefused	
remoteGatekeeperRefused	
remoteGatekeeperUnreachable	
remoteGatewayResources	
gatekeeperForced	
h225SocketError	
h225ProtocolError	
h225DecodeError	
h245SocketError	

Copyright © Codian Ltd. 2004 - 2006

h245ProtocolError	
h245DecodeError	
q931DecodeError	
q931ProtocolError	

7 References

[1]	XML-RPC, http://www.xmlrpc.com/
[2]	RFC 2616, http://www.faqs.org/rfcs/rfc2616.html

Appendix A - Conference Layouts

The conference.participant.add and conference.participant.modify methods allow a specific layout to be specified for video sent to that participant via the “cpLayout” parameter.

The “cpLayout” string parameter can take the following values:

- **default** - use the MCU’s default view family
- **family<index>** - use the specified layout family; see below
- **layout<index>** - use a specific layout; see below

The “<index>” values for “family<index>” correspond to the pane arrangements shown below:

1

2

3

4

5

Copyright © Codian Ltd. 2004 - 2006

The “<index>” values for “layout<index>” correspond to the pane arrangements shown below:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

Copyright © Codian Ltd. 2004 - 2006

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

Copyright © Codian Ltd. 2004 - 2006

Appendix B - Linking conferences across MCUs

For the purposes of this description, two conferences are said to be "linked" if there is a bi-directional H.323 connection between them and each MCU is sending a video channel to the other, showing the active speaker full screen. The audio communicated between the MCUs will be the usual mix of active speakers. For clarification, the linked conferences are given different names (“linked1” and “linked2”) in the below explanation, but it is perfectly possible for the conferences to have the same name.

The first step is to set up the two conferences. The most important part of this is to ensure that the conferences have a numeric id set (the "conferenceID" field in "conference.create"), because without this configured field it is not possible to call in directly to a conference. In the example given here, both conferences are given a numeric id, though strictly it is only necessary to configure this on the *target* MCU (i.e. the one that is called rather than the one calling).

In this specific example, "linked1" is set up on "mcu1" and "linked2" set up on "mcu2". The creation of "linked1" is shown in **example message 1**, and it is configured with numeric id "1234"; the creation of "linked2" is shown in **example message 2**, and this conference is given the numeric id "5678".

Next, a participant needs to be added to the "linked1" conference and connected to "linked2" on the target MCU. The most reliable way to accomplish this, which does not rely on the target MCU's gatekeeper usage, is to call from “mcu1” into the target conference using "mcu2" as a gateway and the target conference's numeric id as the remote address. The participant addition is shown in **example message 3** - as well as the address and gateway, it also configures the view layout to be full screen (by setting "cpLayout" to "layout1"), to make sure that just the active speaker from "linked1" is sent to "linked2".

The final step is slightly more complex - it involves modifying the new "linked2" participant on "mcu2" which was the result of the call from "mcu1". The modification required is to change the view layout setting (for the video sent from "linked2" to "linked1") to full screen so that what gets sent is a view of the “linked2” active speaker.

The complication here is that the "linked2" participant in question is not a participant created via the API, and so the API does not know in advance what name it has. It is thus necessary to poll membership of "linked2" after the connection from "linked1" has been made, identify the participant corresponding to the call, and use its name in a "participant.modify" call to set the view layout. The simplest way to identify the participant is to look for an absence of the “address” field in a “conference.query” response – for incoming, non-API, connections this will not be present. **Example message 4** shows such a “participant.modify” call; in this case the participant name needed was "1_Codian MCU 4210".

B.1 Example message 1 - creating conference "linked1" on "mcu1"

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked1</string>
            </value>
          </member>
```

```
          <member>
            <name>conferenceID</name>
            <value>
              <string>1234</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
<methodCall>
```

B.2 Example message 2 - creating conference "linked2" on "mcu2"

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.create</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked2</string>
            </value>
          </member>
          <member>
            <name>conferenceID</name>
            <value>
              <string>5678</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

B.3 Example message 3 - calling into "linked2" from "linked1"

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.participant.add</methodName>
  <params>
    <param>
      <value>
```

Copyright © Codian Ltd. 2004 - 2006

```
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
              <string>admin</string>
            </value>
          </member>
          <member>
            <name>conferenceName</name>
            <value>
              <string>linked1</string>
            </value>
          </member>
          <member>
            <name>participantName</name>
            <value>
              <string>remote_mcu</string>
            </value>
          </member>
          <member>
            <name>address</name>
            <value>
              <string>5678</string>
            </value>
          </member>
          <member>
            <name>gatewayAddress</name>
            <value>
              <string>10.2.1.27</string>
            </value>
          </member>
          <member>
            <name>cpLayout</name>
            <value>
              <string>layout1</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

B.4 Example message 4 - setting the new "linked2" participant to use a full screen view layout

```
<?xml version="1.0"?>
<methodCall>
  <methodName>conference.participant.modify</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>authenticationUser</name>
            <value>
```

```
            <string>admin</string>
          </value>
        </member>
        <member>
          <name>conferenceName</name>
            <value>
              <string>linked2</string>
            </value>
        </member>
        <member>
          <name>participantName</name>
          <value>
            <string>1_Codian MCU 4210</string>
          </value>
        </member>
        <member>
          <name>cpLayout</name>
          <value>
            <string>layout1</string>
          </value>
        </member>
      </struct>
    </value>
  </param>
 </params>
</methodCall>
```

B.5 Message responses

The response to each of the above method invocations should be the same normal success indication:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>status</name>
            <value>
              <string>operation successful</string>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Copyright © Codian Ltd. 2004 - 2006